PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-244894
(43)Date of publication of application : 30.08.2002

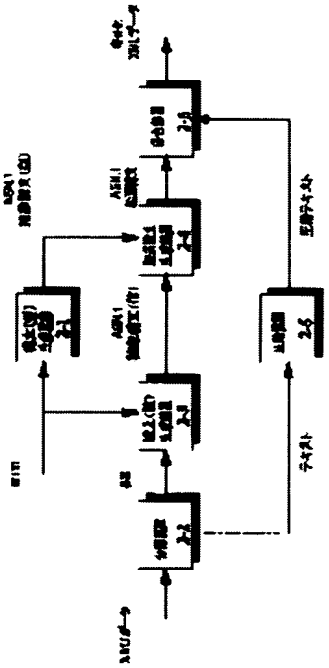
(51)Int.Cl. G06F 12/00
G06F 5/00
G06F 13/00
H03M 7/30
// G06F 17/21

(21)Application number : 2001-027462
(22)Date of filing : 02.02.2001

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : IMAMURA TAKESHI

(54) ENCODING METHOD FOR XML DATA, DECODING METHOD FOR ENCODED XML DATA, ENCODING SYSTEM FOR XML DATA, DECODING SYSTEM FOR ENCODED XML DATA, PROGRAM AND RECORDING MEDIUM

(57)Abstract:
PROBLEM TO BE SOLVED: To enhance the efficiency of encoding (compression) of XML data.
SOLUTION: A syntax (type) generator 2-1 is used to convert a DTD into ASN. 1 abstract syntax (type), and a separator 2-2 is used to separate text from XML data according to the DTD. A syntax (value) generator 2-3 is next used to convert the XML data (syntax of elements) from which the text is separated into ASN.1 abstract syntax (value) according to the ASN.1 abstract syntax (type). A transfer syntax generator 2-4 is then used to convert the ASN.1 abstract syntax (value) into ASN.1 transfer syntax. A compressor 2-5 is used to compress the separated text. A merger 2-6 is further used to merge the ASN.1 transfer syntax and the compressed text to generate encoded XML data.



LEGAL STATUS

[Date of request for examination] 16.10.2001

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2002－244894
（P2002－244894A）
(43)公開日　平成14年８月30日(2002.8.30)

(51)Int.Cl.[7]	識別記号	FI	テーマコード（参考）
G06F 12/00	511	G06F 12/00	511A 5B009
	546		546A 5B082
5/00		5/00	H 5B089
13/00	353	13/00	353C 5J064
H03M 7/30		H03M 7/30	Z

審査請求　有　請求項の数23　ＯＬ　（全 17 頁）　最終頁に続く

(21)出願番号　特願2001－27462(P2001－27462)

(22)出願日　平成13年２月２日(2001.2.2)

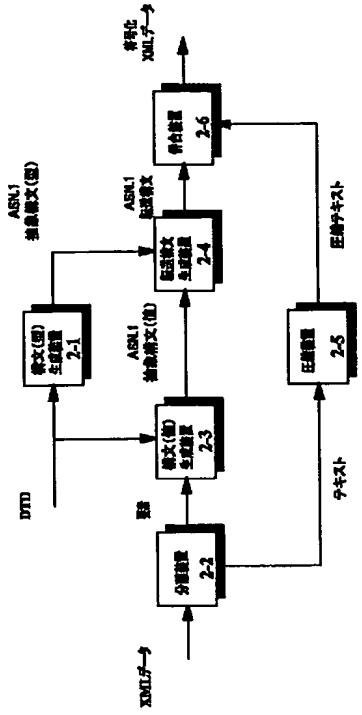
(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504、ニューヨーク州　アーモンク　ニュー　オーチャード　ロード

(74)復代理人　100112520
弁理士　林　茂則　（外３名）

最終頁に続く

(54)【発明の名称】　ＸＭＬデータの符号化方法、符号化されたＸＭＬデータの復号化方法、ＸＭＬデータの符号化システム、符号化されたＸＭＬデータの復号化システム、プログラムおよび記録媒体

(57)【要約】
【課題】　ＸＭＬデータの符号化（圧縮）効率を高める。
【解決手段】　構文（型）生成装置２－１を用いてＤＴＤをＡＳＮ．１抽象構文（型）に変換し、分離装置２－２を用いて前記ＤＴＤに従うＸＭＬデータからテキストを分離する。その後、構文（値）生成装置２－３を用いてテキストが分離された後のＸＭＬデータ（要素の構文）を、前記ＡＳＮ．１抽象構文（型）に従うＡＳＮ．１抽象構文（値）に変換する。その後、転送構文生成装置２－４を用いてＡＳＮ．1抽象構文（値）をＡＳＮ．１転送構文に変換する。一方、圧縮装置２－５を用いて分離されたテキストを圧縮する。さらに、併合装置２－６を用いてＡＳＮ．１転送構文と圧縮テキストを併合し符号化ＸＭＬデータを生成する。



1

【特許請求の範囲】

【請求項１】　ＸＭＬデータの文法を定義する文法定義をＡＳＮ．1抽象構文（型）に変換するステップと、
前記ＸＭＬデータを、要素の内容（テキスト）と構造（構造を含む要素名）とに分離するステップと、
前記構造を、前記ＡＳＮ．１抽象構文（型）に従うＡＳＮ．1抽象構文（値）に変換するステップと、
前記ＡＳＮ．1抽象構文（値）をＡＳＮ．1転送構文に変換するステップと、
前記要素の内容を圧縮するステップと、
圧縮された前記要素の内容と前記ＡＳＮ．1転送構文とを併合するステップと、
を含むＸＭＬデータの符号化方法。

【請求項２】　前記文法定義に属性、処理命令その他前記要素以外の文法定義項目を有し、
前記要素以外の文法定義項目を特別要素として前記要素に含めるように前記文法定義を他の文法定義に変換するステップと、
前記他の文法定義に従うように、前記ＸＭＬデータを他のＸＭＬデータに変換するステップと、
をさらに有する請求項１記載のＸＭＬデータの符号化方法。

【請求項３】　前記文法定義はＤＴＤであり、
前記ＤＴＤの要素内容には、「，」、「｜」、「？」、「＊」または「＋」から選択される単一または複数のオペレータが含まれ、または前記オペレータを含まず、
前記ＡＳＮ．１抽象構文（型）において、
前記「，」オペレータは、ｓｅｑｕｅｎｃｅ型で表現し、
前記「｜」オペレータは、ｃｈｏｉｃｅ型で表現し、
前記「？」オペレータは、ｓｅｑｕｅｎｃｅ型とキーワード「ＯＰＴＩＯＮＡＬ」の組み合わせで表現し、
前記「＊」オペレータは、ｓｅｑｕｅｎｃｅ－ｏｆ型で表現し、
前記「＋」オペレータは、サイズが制限されたｓｅｑｕｅｎｃｅ－ｏｆ型で表現し、
前記オペレータが無いときには、ｄｅｆｉｎｅｄ型で表現する、
請求項１または２記載のＸＭＬデータの符号化方法。

【請求項４】　前記文法定義を他の文法定義に変換するステップにおいて、
前記文法定義に含まれる属性を、一意に決定できる属性要素として表現し、前記属性の親要素の子要素として扱い、
前記属性の属性値は、「ＣＤＡＴＡ」として前記属性要素の子要素として扱い、
前記属性が「ＲＥＱＵＩＲＥＤ」属性の場合、前記属性要素を要素で表現し、
前記属性が「ＩＭＰＬＩＥＤ」属性の場合、または、前記属性にデフォルト値が定義されている場合、前記属性

2

要素を「？」オペレータが適用された要素で表現し、
前記ＸＭＬデータを他のＸＭＬデータに変換するステップにおいて、
前記ＸＭＬデータの要素に含まれる属性を、一意に決定できる属性要素として表現し、前記属性の親要素の子要素として扱う（ただし前記属性にデフォルト値が定義され、かつ前記属性の属性値が前記デフォルト値と一致する場合を除く）、
請求項３記載のＸＭＬデータの符号化方法。

【請求項５】　前記ＡＳＮ．１抽象構文（値）を前記ＡＳＮ．１転送構文に変換するステップにおいて、ＰＥＲ規則を用いる請求項１または２記載のＸＭＬデータの符号化方法。

【請求項６】　ＸＭＬデータの文法を定義する文法定義をＡＳＮ．1抽象構文（型）に変換するステップと、
符号化されたＸＭＬデータを、ＡＳＮ．１転送構文と圧縮された要素の内容（テキスト）とに分離するステップと、
前記ＡＳＮ．１転送構文を、前記ＡＳＮ．１抽象構文（型）に従うＡＳＮ．１抽象構文（値）に変換するステップと、
前記ＡＳＮ．１抽象構文（値）を、前記文法定義に従うＸＭＬデータの構造（構造を含む要素名）に変換するステップと、
前記圧縮された要素の内容を解凍するステップと、
前記解凍された要素の内容と前記ＸＭＬデータの構造とを併合するステップと、
を含む符号化されたＸＭＬデータの復号化方法。

【請求項７】　前記文法定義に属性、処理命令その他前記要素以外の文法定義項目を有し、
前記要素以外の文法定義項目を特別要素として前記要素に含めるように前記文法定義を他の文法定義に変換するステップと、
前記文法定義に従うように、前記復号化されたＸＭＬデータを他のＸＭＬデータに変換するステップと、
をさらに有する請求項６記載の符号化されたＸＭＬデータの復号化方法。

【請求項８】　前記文法定義はＤＴＤであり、
前記ＤＴＤの要素内容には、「，」、「｜」、「？」、「＊」または「＋」から選択される単一または複数のオペレータが含まれ、または前記オペレータを含まず、
前記ＡＳＮ．１抽象構文（型）において、
前記「，」オペレータは、ｓｅｑｕｅｎｃｅ型で表現し、
前記「｜」オペレータは、ｃｈｏｉｃｅ型で表現し、
前記「？」オペレータは、ｓｅｑｕｅｎｃｅ型とキーワード「ＯＰＴＩＯＮＡＬ」の組み合わせで表現し、
前記「＊」オペレータは、ｓｅｑｕｅｎｃｅ－ｏｆ型で表現し、
前記「＋」オペレータは、サイズが制限されたｓｅｑｕ

ｅｎｃｅ－ｏｆ型で表現し、
前記オペレータが無いときには、ｄｅｆｉｎｅｄ型で表現する、
請求項６または７記載の符号化されたＸＭＬデータの復号化方法。

【請求項９】　前記文法定義を他の文法定義に変換するステップにおいて、
前記文法定義に含まれる属性を、一意に決定できる属性要素として表現し前記属性の親要素の子要素として扱い、
前記属性の属性値は、「ＣＤＡＴＡ」として前記属性要素の子要素として扱い、
前記属性が「ＲＥＱＵＩＲＥＤ」属性の場合、前記属性要素を要素で表現し、
前記属性が「ＩＭＰＬＩＥＤ」属性の場合、または、前記属性にデフォルト値が定義されている場合、前記属性要素を「？」オペレータが適用された要素で表現し、
前記復号化されたＸＭＬデータを他のＸＭＬデータに変換するステップにおいて、
前記復号化されたＸＭＬデータの要素に子要素として含まれる前記属性要素を、前記要素の属性またはその属性値に変換する、
請求項８記載の符号化されたＸＭＬデータの復号化方法。

【請求項１０】　前記ＡＳＮ．１転送構文を前記ＡＳＮ．１抽象構文（値）に変換するステップにおいて、ＰＥＲ規則を用いる請求項６または７記載の符号化されたＸＭＬデータの復号化方法。

【請求項１１】　ＸＭＬデータの文法を定義する文法定義をＡＳＮ．1抽象構文（型）に変換する手段と、
前記ＸＭＬデータを、要素の内容（テキスト）と構造（構造を含む要素名）とに分離する手段と、
前記構造を、前記ＡＳＮ．１抽象構文（型）に従うＡＳＮ．1抽象構文（値）に変換する手段と、
前記ＡＳＮ．1抽象構文（値）をＡＳＮ．1転送構文に変換する手段と、
前記要素の内容を圧縮する手段と、
圧縮された前記要素の内容と前記ＡＳＮ．1転送構文とを併合する手段と、
を含むＸＭＬデータの符号化システム。

【請求項１２】　前記文法定義に属性、処理命令その他前記要素以外の文法定義項目を有し、
前記要素以外の文法定義項目を特別要素として前記要素に含めるように前記文法定義を他の文法定義に変換する手段と、
前記他の文法定義に従うように、前記ＸＭＬデータを他のＸＭＬデータに変換する手段と、
をさらに有する請求項１１記載のＸＭＬデータの符号化システム。

【請求項１３】　前記文法定義はＤＴＤであり、

前記ＤＴＤの要素内容には、「，」、「｜」、「？」、「＊」または「＋」から選択される単一または複数のオペレータが含まれ、または前記オペレータを含まず、
前記ＡＳＮ．１抽象構文（型）において、
前記「，」オペレータは、ｓｅｑｕｅｎｃｅ型で表現し、
前記「｜」オペレータは、ｃｈｏｉｃｅ型で表現し、
前記「？」オペレータは、ｓｅｑｕｅｎｃｅ型とキーワード「ＯＰＴＩＯＮＡＬ」の組み合わせで表現し、
前記「＊」オペレータは、ｓｅｑｕｅｎｃｅ－ｏｆ型で表現し、
前記「＋」オペレータは、サイズが制限されたｓｅｑｕｅｎｃｅ－ｏｆ型で表現し、
前記オペレータが無いときには、ｄｅｆｉｎｅｄ型で表現する、
請求項１１または１２記載のＸＭＬデータの符号化システム。

【請求項１４】　前記文法定義を他の文法定義に変換する手段において、
前記文法定義に含まれる属性を、一意に決定できる属性要素として表現し、前記属性の親要素の子要素として扱い、
前記属性の属性値は、「ＣＤＡＴＡ」として前記属性要素の子要素として扱い、
前記属性が「ＲＥＱＵＩＲＥＤ」属性の場合、前記属性要素を要素で表現し、
前記属性が「ＩＭＰＬＩＥＤ」属性の場合、または、前記属性にデフォルト値が定義されている場合、前記属性要素を「？」オペレータが適用された要素で表現し、
前記ＸＭＬデータを他のＸＭＬデータに変換する手段において、
前記ＸＭＬデータの要素に含まれる属性を、一意に決定できる属性要素として表現し、前記属性の親要素の子要素として扱う（ただし前記属性にデフォルト値が定義され、かつ前記属性の属性値が前記デフォルト値と一致する場合を除く）、
請求項１３記載のＸＭＬデータの符号化システム。

【請求項１５】　前記ＡＳＮ．１抽象構文（値）を前記ＡＳＮ．１転送構文に変換する手段において、ＰＥＲ規則を用いる請求項１１または１２記載のＸＭＬデータの符号化システム。

【請求項１６】　ＸＭＬデータの文法を定義する文法定義をＡＳＮ．1抽象構文（型）に変換する手段と、
符号化されたＸＭＬデータを、ＡＳＮ．１転送構文と圧縮された要素の内容（テキスト）とに分離する手段と、
前記ＡＳＮ．１転送構文を、前記ＡＳＮ．１抽象構文（型）に従うＡＳＮ．１抽象構文（値）に変換する手段と、
前記ＡＳＮ．１抽象構文（値）を、前記文法定義に従うＸＭＬデータの構造（構造を含む要素名）に変換する手

段と、
前記圧縮された要素の内容を解凍する手段と、
前記解凍された要素の内容と前記ＸＭＬデータの構造とを併合する手段と、
を含む符号化されたＸＭＬデータの復号化システム。
【請求項１７】　前記文法定義に属性、処理命令その他前記要素以外の文法定義項目を有し、
前記要素以外の文法定義項目を特別要素として前記要素に含めるように前記文法定義を他の文法定義に変換する手段と、
前記文法定義に従うように、前記復号化されたＸＭＬデータを他のＸＭＬデータに変換する手段と、
をさらに有する請求項１６記載の符号化されたＸＭＬデータの復号化システム。
【請求項１８】　前記文法定義はＤＴＤであり、
前記ＤＴＤの要素内容には、「，」、「｜」、「？」、「＊」または「＋」から選択される単一または複数のオペレータが含まれ、または前記オペレータを含まず、
前記ＡＳＮ．１抽象構文（型）において、
前記「，」オペレータは、ｓｅｑｕｅｎｃｅ型で表現し、
前記「｜」オペレータは、ｃｈｏｉｃｅ型で表現し、
前記「？」オペレータは、ｓｅｑｕｅｎｃｅ型とキーワード「ＯＰＴＩＯＮＡＬ」の組み合わせで表現し、
前記「＊」オペレータは、ｓｅｑｕｅｎｃｅ－ｏｆ型で表現し、
前記「＋」オペレータは、サイズが制限されたｓｅｑｕｅｎｃｅ－ｏｆ型で表現し、
前記オペレータが無いときには、ｄｅｆｉｎｅｄ型で表現する、
請求項１６または１７記載の符号化されたＸＭＬデータの復号化システム。
【請求項１９】　前記文法定義を他の文法定義に変換する手段において、
前記文法定義に含まれる属性を、一意に決定できる属性要素として表現し、前記属性の親要素の子要素として扱い、
前記属性の属性値は、「ＣＤＡＴＡ」として前記属性要素の子要素として扱い、
前記属性が「ＲＥＱＵＩＲＥＤ」属性の場合、前記属性要素を要素で表現し、
前記属性が「ＩＭＰＬＩＥＤ」属性の場合、または、前記属性にデフォルト値が定義されている場合、前記属性要素を「？」オペレータが適用された要素で表現し、
前記復号化されたＸＭＬデータを他のＸＭＬデータに変換する手段において、
前記復号化されたＸＭＬデータの要素に子要素として含まれる前記属性要素を、前記要素の属性またはその属性値に変換する、
請求項１８記載の符号化されたＸＭＬデータの復号化システム。
【請求項２０】　前記ＡＳＮ．１転送構文を前記ＡＳＮ．１抽象構文（値）に変換する手段において、ＰＥＲ規則を用いる請求項１６または１７記載の符号化されたＸＭＬデータの復号化システム。
【請求項２１】　コンピュータが実行可能なプログラムであって、コンピュータに、
ＸＭＬデータの文法を定義する文法定義をＡＳＮ．1抽象構文（型）に変換する機能と、
前記ＸＭＬデータを、要素の内容（テキスト）と構造（構造を含む要素名）とに分離する機能と、
前記構造を、前記ＡＳＮ．１抽象構文（型）に従うＡＳＮ．1抽象構文（値）に変換する機能と、
前記ＡＳＮ．1抽象構文（値）をＡＳＮ．1転送構文に変換する機能と、
前記要素の内容を圧縮する機能と、
圧縮された前記要素の内容と前記ＡＳＮ．1転送構文とを併合する機能と、
を実現させるためのプログラム。
【請求項２２】　コンピュータが実行可能なプログラムであって、コンピュータに、
ＸＭＬデータの文法を定義する文法定義をＡＳＮ．1抽象構文（型）に変換する機能と、
符号化されたＸＭＬデータを、ＡＳＮ．１転送構文と圧縮された要素の内容（テキスト）とに分離する機能と、
前記ＡＳＮ．１転送構文を、前記ＡＳＮ．１抽象構文（型）に従うＡＳＮ．１抽象構文（値）に変換する機能と、
前記ＡＳＮ．１抽象構文（値）を、前記文法定義に従うＸＭＬデータの構造（構造を含む要素名）に変換する機能と、
前記圧縮された要素の内容を解凍する機能と、
前記解凍された要素の内容と前記ＸＭＬデータの構造とを併合する機能と、
を実現させるためのプログラム。
【請求項２３】　コンピュータ読取可能な記録媒体であって、
コンピュータに、ＸＭＬデータの文法を定義する文法定義をＡＳＮ．1抽象構文（型）に変換する機能と、前記ＸＭＬデータを、要素の内容（テキスト）と構造（構造を含む要素名）とに分離する機能と、前記構造を、前記ＡＳＮ．１抽象構文（型）に従うＡＳＮ．1抽象構文（値）に変換する機能と、前記ＡＳＮ．1抽象構文（値）をＡＳＮ．1転送構文に変換する機能と、前記要素の内容を圧縮する機能と、圧縮された前記要素の内容と前記ＡＳＮ．1転送構文とを併合する機能と、を実現させるための第１プログラム、または、
コンピュータに、ＸＭＬデータの文法を定義する文法定義をＡＳＮ．1抽象構文（型）に変換する機能と、符号化されたＸＭＬデータを、ＡＳＮ．１転送構文と圧縮さ

れた要素の内容（テキスト）とに分離する機能と、前記ＡＳＮ．１転送構文を、前記ＡＳＮ．１抽象構文（型）に従うＡＳＮ．１抽象構文（値）に変換する機能と、前記ＡＳＮ．１抽象構文（値）を、前記文法定義に従うＸＭＬデータの構造（構造を含む要素名）に変換する機能と、前記圧縮された要素の内容を解凍する機能と、前記解凍された要素の内容と前記ＸＭＬデータの構造とを併合する機能と、を実現させるための第２プログラム、の何れかのプログラムが記録された記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、データの符号化および復号化技術に関し、特にＸＭＬ(extensiblemarkup language)データの符号化における圧縮率の向上に適用して有効な技術に関する。

【０００２】

【従来の技術】近年、インターネット上でのデータ表現の手段としてＸＭＬが注目されている。ＸＭＬは拡張可能なメタ言語であり、ユーザが独自に文法を規定することができる。また、各要素に論理的な意味を持たせることが可能であり、ＨＴＭＬ(hypertext markup language)に比較してデータ処理が大幅に容易になる。このため、電子商取引等インターネットで交換される構造化文書の標準的な表現方式として期待されている。なお、ＸＭＬについては、「W3C. Extensible Markup Language (XML) 1.0, 1998. http://www.w3.org/TR/REC-xml」に詳細が記述されている。

【０００３】ＸＭＬデータは文字で記述されるため可読性が高い利点がある。その反面、冗長性も高いという難点がある。すなわち要素の意味は主に開始タグ内に記述され、その意味内容は文字で記述されている内容を参照すれば容易に把握できる。しかし、全てが文字で記述されるため、全体の文字数が増加し、ＸＭＬデータ全体の情報量（文字数）が多くなる。文字数が多ければ、たとえばストレージに格納したり、あるいはネットワークで転送する際に、記録容量あるいは転送量が多くなり物理的、時間的コストの上昇を来たす。そこで、ＸＭＬデータを短い符号に符号化（または圧縮）できれば都合がよい。

【０００４】データの圧縮法には各種の手法が知られている。たとえば、ランレングス符号、Ｈｕｆｆｍａｎ符号、算術符号、ＬＺ７７等である。これら符号化の手法については、たとえば、「Huffman, D.A. “A method for the construction of minimum-redundancy codes” Proc. of the IRE September, 1952」、「Mark Nelsonand Jean Loup Gailly “The Data Compression Book”, Second Edition. M&TBooks 1996」、「Jacob Ziv and Abraham Lempel. “A universal algorithm for sequential data compression” IEEE Transactions on Information TheoryMay, 1977」に詳しい。

【０００５】しかしながら、これら圧縮手法はＸＭＬに特化されたものではなく、ＸＭＬデータに適用した場合に必ずしも圧縮効率が高いものではない。ＸＭＬデータに特化した圧縮手法には、たとえば「D. Suciu and H. Liefke. XMill: an Efficient Compressor for XML Data, 1999. ｈｔｔｐ：／／ｗｗｗ．ｒｅｓｅａｒｃｈ．ａｔｔ．ｃｏｍ／ｓｗ／ｔｏｏｌｓ／ｘｍｉｌｌ／」に記載のＸＭｉｌｌ、「XML Solutions Corp. XML Zip, 1999. ｈｔｔｐ：／／ｗｗｗ．ｘｍｌｓ．ｃｏｍ／ｐｒｏｄｕｃｔｓ／ｘｍｌｚｉｐ／ｘｍｌｚｉｐ．ｈｔｍｌ」に記載のＸＭＬＺｉｐ、あるいは「井川甲作著、東京工業大学工学部情報工学科卒業論文「ＤＴＤを用いたＸＭＬ文書圧縮アルゴリズムに関する研究」、平成１２年２月」に記載のＸＣｏｍｐなどがある。

【０００６】ＸＭｉｌｌは、ＸＭＬデータから要素ごとのコンテンツ（テキスト）部分を抽出する。この抽出された部分をコンテナと呼ぶ。そして構造部分を数字で符号化し、テキスト部分はコンテナごとに　ＬＺ７７などの方法で圧縮する。基本的にはパラメータ等の情報を必要とせず、アプリケーションのみで圧縮が可能である。必要であればパラメータ等を指定してコンテナごとの圧縮方法を指定し、圧縮効率を高めることが可能である。また、Ｃで実装されるため圧縮速度が速いという特徴をもつ。

【０００７】ＸＭＬＺｉｐは、ルート要素からの深さを指定し、指定部分をドキュメント要素から分割し、残りをＺＩＰで圧縮する。ルート要素部分は符号化されず直接操作することが可能になる。使用しない部分を圧縮することで文書へのアクセスを迅速に行える。ただし、圧縮効率はＸＭｉｌｌに比較して低い。

【０００８】ＸＣｏｍｐは、ＸＭＬデータの構造部分のうち、ＤＴＤ(Document Type Definition)から一意に決定される部分は符号化せず、一意に決定できない部分についての構造部分のみを圧縮する。テキスト部分はＸＭｉｌｌ　と同じ方法で圧縮する。すなわち、以下の手順で圧縮を行う。（１）構造とコンテンツを分離する。（２）ＤＴＤからプッシュダウンオートマトン（ＰＤＡ）を生成する。（３）生成されたＰＤＡを用いて構造部分を符号化する符号化トランスデューサを生成する。（４）符号化トランスデューサの各ノードに割り振られた数字をオートマトンを連鎖的に遷移することにより出力し、構造を符号化する。（５）得られた構造符号と要素ごとのコンテンツをＬＺ７７等で圧縮し、圧縮されたＸＭＬ文書を出力する。

【０００９】上記ＸＭＬ文書に特化された圧縮手法のうち、ＸＣｏｍｐは構造部分の一部を符号化しないので、かなりよい圧縮効率を達成する。

【００１０】

【発明が解決しようとする課題】上記の通り、ＸＣｏｍｐはＸＭＬ文書の圧縮手法として相対的に優れる。しか

し、本発明者の検討によればＸＣｏｍｐにおいても、ＸＭＬデータがある特定の構造をしている場合に圧縮効率が悪くなるという課題がある。すなわち、要素に「？」オペレータや「＊」オペレータ（「＋」オペレータを含む）が適用された場合である。

【００１１】「？」オペレータは、ある要素の子要素が０回もしくは１回出現する場合に、要素の宣言文において子要素に付されるオペレータである。ＸＣｏｍｐにおいて「？」オペレータが出現した場合、「？」オペレータは、ある状態から別の状態に遷移する複数の選択肢で表現される。選択肢にはインデクスが付与される。ＸＣｏｍｐの実行時には、選択された選択肢に付与されたインデクスが出力される。選択肢の数は、「？」オペレータの後に「？」オペレータなどがいくつ連続するかによる。例えば、「？」オペレータがｎ個連続する場合に、最初の「？」オペレータに対する選択肢はｎ＋１個となる。従って、インデクスはｎ＋１種類必要であり、その１つを表現するにはＯ（ｌｏｇ　ｎ）ビット必要である。「？」オペレータが適用された要素がすべて存在すると、インデクスが複数羅列される。その１つを表現するのにＯ（ｌｏｇ　ｎ）ビット必要であるから、それ全体を表現するにはＯ（ｎｌｏｇ　ｎ）ビット必要である。

【００１２】「＊」オペレータは、ある要素の子要素が０回以上出現する場合に、要素の宣言文において子要素に付されるオペレータであり、「＋」オペレータは、ある要素の子要素が１回以上出現する場合に、要素の宣言文において子要素に付されるオペレータである。ＸＣｏｍｐにおいて「＊」オペレータ（または「＋」オペレータ）が出現した場合、「＊（または＋）」オペレータは、同じ状態か別の状態に遷移する２つの選択肢で表現される。選択肢には、インデクスが付与される。ＸＣｏｍｐの実行時には、選択された選択肢に付与されたインデクスが出力される。「＊」オペレータが適用された要素が複数存在すると、同じインデクスが複数羅列される。インデクスの数は、存在する要素の数に比例する。従って、インデクス全体を表現するのに、要素の数をｎとすると、Ｏ（ｎ）ビット必要である。

【００１３】すなわち、ＸＣｏｍｐではＤＴＤから一意的に決まらない構造のうち特定のものについて、それらを表現する符号のビット数が大きくなって、必ずしも十分な圧縮効率を達成できない問題がある。

【００１４】本発明の目的は、より圧縮効率の高いＸＭＬデータ（ＸＭＬ文書）の符号化方式を提案し、それを実現する方法およびシステムを提供することにある。

【００１５】

【課題を解決するための手段】より高いＸＭＬデータの圧縮効率を達成するために、本発明では、「ITU-T. X.680 - Abstract Syntax Notation One (ASN.1): Specification of basic notation, 1997. http://www.itu.int/itudoc/itu-t/rec/x/x500up/x680.html」に記載のＡＳＮ．１を利用する。すなわち、符号化対象のＸＭＬデータを構造とコンテンツ（テキスト）に分離する。一方、符号化対象のＸＭＬデータの文法をＡＳＮ．１抽象構文（型）に変換する。分離したコンテンツは要素ごとあるいはまとめてＸＭｉｌｌと同様に圧縮する。また、分離した構造（要素）をＡＳＮ．１抽象構文（型）に従うＡＳＮ．１抽象構文（値）に変換する。次に、ＡＳＮ．１抽象構文（値）をＡＳＮ．１で規定されている符号化規則を用いてＡＳＮ．１転送構文に変換する。ＡＳＮ．１で規定される符号化規則には、ＢＥＲ、ＤＥＲ、ＰＥＲ等があるが、特に符号化効率の観点からＰＥＲを用いることが好ましい。そして符号化された要素（構文）であるＡＳＮ．１転送構文と圧縮されたコンテンツ（テキスト）を併合し、符号化されたＸＭＬデータを生成する。なお、ＰＥＲについては、「ITU-T. X.691-ASN.1 encoding rules: Specification of PackedEncoding Rules (PER)、1997. http://www.itu.int/itudoc/itu-t/rec/x/x500up/x691.html」に詳しく記載されている。

【００１６】このような符号化手法を用いることにより、ＸＣｏｍｐに比較して圧縮効率の高い符号化が行える。たとえば「？」オペレータがｎ個連続して出現した場合の構造の符号化では、本発明を用いることによりＯ（ｎ）ビットでそれを表現できる。また、「＊」オペレータが出現した場合の構造の符号化では、本発明を用いることによりオーダとしてはＸＣｏｍｐと同じであるが、実質的には少ないビット数でそれを表現できる。そして符号化（圧縮）されたＸＭＬデータを通信あるいはストレージに用いることにより通信負荷を低減し、またストレージの容量を節約できる。

【００１７】なお、符号化されたＸＭＬデータの復号化は、前記と逆の処理を行うことにより実現できる。また、本発明の符号化あるいは復号化の方法は、システムとして把握することも可能であり、さらに、前記方法によって実現される機能をコンピュータに実現させるためのプログラムとして把握することも可能である。

【００１８】

【発明の実施の形態】以下、本発明の実施の形態を図面を用いて詳細に説明する。ただし、本発明は多くの異なる態様で実施することが可能であり、本実施の形態の記載内容に限定して解釈すべきではない。なお、実施の形態の全体を通して同じ要素には同じ番号を付するものとする。

【００１９】以下の実施の形態では、主に方法またはシステムについて説明するが、当業者であれば明らかなとおり、本発明はコンピュータで使用可能なプログラムとしても実施できる。したがって、本発明は、ハードウェアとしての実施形態、ソフトウェアとしての実施形態またはソフトウェアとハードウェアとの組合せの実施形態をとることができる。プログラムは、ハードディスク、

ＣＤ－ＲＯＭ、光記憶装置または磁気記憶装置等の任意のコンピュータ可読媒体に記録できる。

【００２０】また以下の実施の形態では、一般的なコンピュータシステムを用いることができる。実施の形態で用いることができるコンピュータシステムは、中央演算処理装置（ＣＰＵ）、主記憶装置（メインメモリ：ＲＡＭ）、不揮発性記憶装置（ＲＯＭ）、コプロセッサ、画像アクセラレータ、キャッシュメモリ、入出力制御装置（Ｉ／Ｏ）等、一般的にコンピュータシステムに備えられるハードウェア資源を備える。また、ハードディスク装置等の外部記憶装置、インターネット等のネットワークに接続可能な通信手段を備えることができる。コンピュータシステムには、パーソナルコンピュータ、ワークステーション、メインフレームコンピュータ等各種のコンピュータが含まれる。

【００２１】（実施の形態１）

１．前提条件

本実施の形態を説明するにあたり、本実施の形態における前提となる条件を述べる。

（１）ＸＭＬデータは、文法で構造が定義される。また、文法は、ＸＭＬデータを符号化する側と復号化する側で共有される。たとえば文法定義ファイルを外部データとして所定のＩＰアドレスに記録し、符号化あるいは復号化する際にＩＰアドレスを用いてこの外部ファイルを参照できる。なお、本実施の形態では文法定義としてＤＴＤを用いるが、これに限られない。例えば、ＸＭＬ　ＳｃｈｅｍａやＲＥＬＡＸなどを用いることができる。

（２）ＸＭＬデータは、要素とテキストだけから構成される。その他のもの（例えば、属性や処理命令）は、別の方法で管理される。例えば、属性や処理命令を特別な要素として表現し、ＸＭＬデータの中に埋め込むことができる。この際、文法も変更する必要がある。また、ＸＭＬデータから分離して、ＸＰｏｉｎｔｅｒなどと共に別に格納することも考えられる。

（３）テキストは、ＸＭｉｌｌなどの方法で圧縮される。このような前提条件の下に本実施の形態のＸＭＬデータの符号化および復号化が行われることを説明する。

【００２２】２．符号化手順

２．１　システム構成と手順の概要

図１は本実施の形態の符号化システムの一例をその機能について示したブロック図である。本実施の形態の符号化システムは、構文（型）生成装置２－１、分離装置２－２、構文（値）生成装置２－３、転送構文生成装置２－４、圧縮装置２－５、併合装置２－６を有する。

【００２３】構文（型）生成装置２－１は、ＤＴＤからＡＳＮ．１抽象構文（型）を生成し、分離装置２－２は、ＸＭＬデータから要素の内容（テキスト）と構造（要素名および構造）を分離する。構文（値）生成装置２－３は、要素の構造からＡＳＮ．１抽象構文（値）を生成し、転送構文生成装置２－４は、ＡＳＮ．１転送構文を生成する。圧縮装置２－５は分離したテキストを圧縮し、併合装置２－６は圧縮テキストとＡＳＮ．１転送構文を併合して符号化されたＸＭＬデータを生成する。

【００２４】このような符号化システムを用いた本実施の形態の符号化方法の手順の概要は以下のとおりである。

（ステップ２－１）　構文（型）生成装置２－１を用いてＤＴＤをＡＳＮ．１抽象構文（型）に変換する。

（ステップ２－２）　分離装置２－２を用いてステップ２－１のＤＴＤに従うＸＭＬデータからテキストを分離する。なお、ステップ２－２は、ステップ２－１と並行に行われても良く、ステップ２－１よりも先に行われても良い。

（ステップ２－３）　構文（値）生成装置２－３を用いてステップ２－２でテキストが分離された後のＸＭＬデータ（要素の構文）を、ステップ２－１のＡＳＮ．１抽象構文（型）に従うＡＳＮ．１抽象構文（値）に変換する。

（ステップ２－４）　転送構文生成装置２－４を用いてステップ２－３のＡＳＮ．１抽象構文（値）をＡＳＮ．１転送構文に変換する。

（ステップ２－５）　圧縮装置２－５を用いてステップ２－２で分離されたテキストを圧縮する。なお、ステップ２－５はステップ２－３～ステップ２－４と並行に行われる。

（ステップ２－６）　併合装置２－６を用いてステップ２－４で生成されたＡＳＮ．１転送構文とステップ２－５で生成された圧縮テキストを併合し、符号化ＸＭＬデータを生成する。

【００２５】なお、ＡＳＮ．１抽象構文（値）をＡＳＮ．１転送構文に変換するには、ＡＳＮ．１で規定されている符号化規則に従う。そのような規則には、ＢＥＲやＤＥＲ、ＰＥＲなどがある。特に、ＰＥＲは、ＡＳＮ．１抽象構文（型）から一意に決定される型やその値などを符号化しないため、符号化効率がよい。

【００２６】以下、前記各ステップを詳細に説明する。

２．２　ステップ２－１

ステップ２－１では、ＤＴＤをＡＳＮ．１抽象構文（型）に変換する。以下では、内容モデルのパターンごとに、その方法を説明する。

【００２７】２．２．１　要素内容(element content)

ＸＭＬの要素内容は、要素名とオペレータの組み合わせで構成される。通常要素名は文字列で表される。オペレータは要素内容における子要素の出現順と出現回数を指定するための演算子である。周知のとおりＸＭＬでは、「，」、「｜」、「？」、「＊」、「＋」の各オペレータが許可される。原則として、要素名はＡＳＮ．１抽象構文では識別子として、オペレータは同じくＡＳＮ．１抽象構文では型として表現する。ただし、識別子はＢＥ

Ｒなどでは符号化されないため、要素名を識別子とする必然性はない。以下では、オペレータごとに、それをどのような型で表現するかを説明する。

【００２８】２．２．１．１　「，」オペレータ

「，」オペレータは、ＡＳＮ．１抽象構文ではｓｅｑｕｅｎｃｅ型で表現する。例えば、ＤＴＤとして、

```
<!ELEMENT a (b,c)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (#PCDATA)>
```

が与えられるとき、ＡＳＮ．１抽象構文（型）は、

```
A ::= SEQUENCE {

    b   B,
    c   C }
B ::= NULL
C ::= NULL
```

となる。ここで、Ａ、Ｂ、Ｃは、便宜上導入した型参照である。型参照は、衝突さえしなければどのようなものでもよい。また、テキストはステップ２－２で分離されるため、ＢやＣはｎｕｌｌ型とする。

【００２９】２．２．１．２　「｜」オペレータ

「｜」オペレータは、ＡＳＮ．１抽象構文ではｃｈｏｉｃｅ型で表現する。例えば、ＤＴＤとして、

```
<!ELEMENT a (b|c)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c(#PCDATA)>
```

が与えられているとき、ＡＳＮ．１抽象構文（型）は、

```
A ::= CHOICE {
    b   B,
    c   C }
B ::= NULL
C ::= NULL
```

となる。

【００３０】２．２．１．３　「？」オペレータ

「？」オペレータは、ＡＳＮ．１抽象構文ではｓｅｑｕｅｎｃｅ型とキーワード「ＯＰＴＩＯＮＡＬ」の組み合わせで表現する。例えば、ＤＴＤとして、

```
<!ELEMENT a (b?)>
<!ELEMENT b (#PCDATA)>
```

が与えらるとき、ＡＳＮ．１抽象構文（型）は、

```
A ::= SEQUENCE {

    b   B OPTIONAL }
B ::= NULL
```

となる。

【００３１】２．２．１．４　「＊」オペレータ

「＊」オペレータは、ＡＳＮ．１抽象構文ではｓｅｑｕｅｎｃｅ－ｏｆ型で表現する。例えば、ＤＴＤとして、

```
<!ELEMENT a (b*)>
<!ELEMENT b (#PCDATA)>
```


が与えられるとき、ＡＳＮ．１抽象構文（型）は、

```
A ::= SEQUENCE OF B
B ::= NULL
```

となる。

【００３２】２．２．１．５　「＋」オペレータ

「＋」オペレータは、ＡＳＮ．１抽象構文ではサイズが制限されたｓｅｑｕｅｎｃｅ－ｏｆ型で表現する。ただし、サイズ制限は符号化には影響しないため、「＋」オペレータに対する符号は「＊」オペレータに対するそれと同じものになる。例えば、ＤＴＤとして、

```
<!ELEMENT a (b+)>
<!ELEMENT b (#PCDATA)>
```

が与えられるとき、ＡＳＮ．１抽象構文（型）は、

```
A ::= SEQUENCE SIZE (1 .. MAX) OF B
B ::= NULL
```

となる。

【００３３】２．２．１．６　オペレータなし

要素内容には、オペレータの適用されていない要素名が１つだけ指定されることもある。そのような要素名は、ＡＳＮ．１抽象構文ではｄｅｆｉｎｅｄ型で表現する。例えば、ＤＴＤとして、

```
<!ELEMENT a (b)>
<!ELEMENT b (#PCDATA)>
```

が与えられるとき、ＡＳＮ．１抽象構文（型）は、

```
A ::= B
B ::= NULL
```

となる。

【００３４】２．２．２　混在内容(mixed content)

混在内容は、キーワード「＃ＰＣＤＡＴＡ」と１つ以上の要素名を「｜」オペレータで結合した後、「＊」オペレータを適用したものである。そこで、ＡＳＮ．１抽象構文ではｃｈｏｉｃｅ型とｓｅｑｕｅｎｃｅ－ｏｆ型の組み合わせで表現する。例えば、ＤＴＤとして、

```
<!ELEMENT a (#PCDATA|b)*>
<!ELEMENT b (#PCDATA)>
```

が与ええられるとき、ＡＳＮ．１抽象構文（型）は、

```
A ::= SEQUENCE OF CHOICE {
    txt   NULL,
    b     B }
B :: NULL
```

となる。ここで、ｔｘｔは便宜上導入した識別子であり、混在内容に含まれるテキストに対応する。

【００３５】２．２．３　空要素(EMPTY)

空要素は、ＡＳＮ．１抽象構文ではｎｕｌｌ型で表現する。例えば、ＤＴＤが、

```
<!ELEMENT a EMPTY>
```

の場合はＡＳＮ．１抽象構文（型）は、

```
A ::= NULL
```

となる。

【００３６】２．２．４　任意要素(ANY)

任意要素は、キーワード「＃ＰＣＤＡＴＡ」とＤＴＤで

15

宣言されたすべての要素名から構成される混在内容と等価である。従って、任意要素のＡＳＮ．１抽象構文における表現は、混在内容のそれに帰着される。

【００３７】２．３　ステップ２－２

ステップ２－２では、ステップ２－１のＤＴＤに従うＸＭＬデータからテキストを分離する。例えば、ＸＭＬデータが、

```
〈a〉
  〈b〉10〈/b〉
  〈c〉20〈/c〉
〈/a〉
```

の場合、要素ｂから「１０」が、要素ｃから「２０」が分離される。その結果、ＸＭＬデータ（要素名および構造）は次のようになる。

```
〈a〉
  〈b /〉
  〈c /〉
〈/a〉
```

【００３８】分離されたテキストを圧縮する方法は、例えば、ＸＭｉｌｌなどのように、要素ごとにまとめた後、圧縮することができる。なお、その他の圧縮方法を適用してももちろん良い。

【００３９】２．４　ステップ２－３

ステップ２－３では、ステップ２－２のＸＭＬデータを、ステップ２－１のＡＳＮ．１抽象構文（型）に従うＡＳＮ．１抽象構文（値）に変換する。例えば、ＡＳＮ．１抽象構文（型）が、

```
A ::= SEQUENCE {
  b  B,
  c  C }
B ::= NULL
C ::= NULL
```

の場合、以下のＸＭＬデータ（要素名および構造）、

```
〈a〉
  〈b /〉
  〈c /〉
〈/a〉
```

をＡＳＮ．１抽象構文（値）に変換すると、

```
a A ::= {
  b  NULL,
  c  NULL }
```

となる。

【００４０】２．５　ステップ２－４

ステップ２－４では、ＡＳＮ．１で規定されている符号化規則に従って、ステップ２－３のＡＳＮ．１抽象構文（値）をＡＳＮ．１転送構文に変換する。そのような規則にはＢＥＲやＤＥＲ、ＰＥＲ（ＡＬＩＧＮＥＤ／ＵＮＡＬＩＧＮＥＤ）などがある。しかしながら、符号化効率を向上する観点からＰＥＲ（ＵＮＡＬＩＧＮＥＤ）を

16

利用することが好ましい。尤も、ＢＥＲ、ＤＥＲ、ＰＥＲ（ＡＬＩＧＮＥＤ）を利用してもよいことは勿論である。

【００４１】２．６　ステップ２－５

ステップ２－５では、テキストを圧縮する。圧縮の具体的方法については、周知のＬＺ７７等を用いる。その他、従来技術の項で説明した公知技術を用いることができる。圧縮は、要素ごとに行われても良く、また、各要素をまとめて圧縮しても良い。

【００４２】２．７　ステップ２－６

ステップ２－６では、ＡＳＮ．１転送構文と圧縮テキストを併合する。両データを単に結合することも可能であるが、復号時の分離を考慮したセパレータをデータ間に挿入し、あるいはデータビット数情報を持つヘッダ等を付加しても良い。

【００４３】３．復号化手順

３．１　システム構成と手順の概要

図２は本実施の形態の復号化システムの一例をその機能について示したブロック図である。本実施の形態の復号化システムは、構文（型）生成装置３－１、転送構文復号装置３－２、抽象構文復号装置３－３、併合装置３－４、分離装置３－５、解凍装置３－６を有する。

【００４４】構文（型）生成装置３－１は、前記した構文（型）生成装置２－１と同様に、ＤＴＤをＡＳＮ．１抽象構文（型）に変換する。分離装置３－５は、符号化されたＸＭＬデータからＡＳＮ．１転送構文と圧縮テキストを分離し、転送構文復号装置３－２は、ＡＳＮ．１転送構文をＡＳＮ．１抽象構文（型）に従うＡＳＮ．１抽象構文（値）に変換する。抽象構文復号装置３－３は、ＡＳＮ．１抽象構文（値）をＤＴＤに従うＸＭＬデータ（要素名と構造）に変換する。併合装置３－４は復号化されたテキスト（要素の内容）とＸＭＬデータ（要素名と構造）を併合し、ＸＭＬデータを生成する。解凍装置３－６は、圧縮テキストを解凍する。

【００４５】このような復号化システムを用いた本実施の形態の復号化方法の手順は、前記符号化の手順をほぼ逆に行う。その概要は以下のとおりである。

（ステップ３－１）　ＤＴＤをＡＳＮ．１抽象構文（型）に変換する。

（ステップ３－２）　符号化ＸＭＬデータを圧縮テキストとＡＳＮ．１転送構文に分離する。なお、ステップ３－２は、ステップ３－１と並行に行われても良く、ステップ３－１よりも先に行われても良い。

（ステップ３－３）　ＡＳＮ．１転送構文を、ステップ３－１のＡＳＮ．１抽象構文（型）に従うＡＳＮ．１抽象構文（値）に変換する。

（ステップ３－４）　ステップ３－３のＡＳＮ．１抽象構文（値）を、ステップ３－１のＤＴＤに従うＸＭＬデータ（要素名と構造）に変換する。

（ステップ３－５）　ステップ３－２で分離した圧縮テ

キストを解凍する。

（ステップ３－６）　ステップ３－４のＸＭＬデータ（要素名と構造）にステップ３－５で解凍したテキストを結合する。

【００４６】なお、前記ステップ３－１～ステップ３－６における復号化の各処理は、それに相当する符号化の場合の逆であって自明である。よって、その詳細な説明を省略する。以下、ＤＴＤの変換およびそのＤＴＤに従うＸＭＬデータのＰＥＲによる符号化の具体例を示す。

【００４７】４．「，」オペレータを含む場合

ここでは、「，」オペレータを含むＤＴＤと、それに従うＸＭＬデータを例にとる。以下のＤＴＤ、

```
<!ELEMENT a (b,c)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (#PCDATA)>
```

が与えられている場合、前記ステップ２－１で生成されるＡＳＮ．１抽象構文（型）は、

```
A ::= SEQUENCE {
   b   B,
   c   C }
B ::= NULL
C ::= NULL
```

となる。このようなＤＴＤに従う符号化対象のＸＭＬデータとして、

```
〈a〉
   〈b〉10〈/b〉
   〈c〉20〈/c〉
〈/a〉
```

を例示すれば、前記ステップ２－２で分離される要素（ＸＭＬデータの要素名と構造）は、

```
〈a〉
   〈b /〉
   〈c /〉
〈/a〉
```

となる。この要素から前記ステップ２－３で生成されるＡＳＮ．１抽象構文（値）は、

```
a A ::= {
   b   NULL,
   c   NULL }
```

となる。

【００４８】ＰＥＲでは、ｓｅｑｕｅｎｃｅ型の値は、原則としてその構成要素の値がその順番で符号化される。しかし、ここではｂやｃの値はｎｕｌｌであり、ｎｕｌｌは空ビット列に符号化されるため、ａの値は空ビット列となる。その場合に、符号は例外的に、

$00000000_{(2)}$

のようになる。このようにして、ＡＳＮ．１転送構文が生成される。なお、このようなＡＳＮ．１転送構文の生成は前記ステップ２－４で行われる。また、下付き文字

の“（２）”は２進数であることを示す。

【００４９】その後、前記ステップ２－５で圧縮処理された圧縮テキストとＡＳＮ．１転送構文（ここでは「$00000000_{(2)}$」）が前記ステップ２－６で併合され、符号化ＸＭＬデータが生成される。

【００５０】５．「｜」オペレータを含む場合ここでは、「｜」オペレータを含むＤＴＤと、それに従うＸＭＬデータを例にとる。以下のＤＴＤ、

```
<!ELEMENT a (b|c)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c(#PCDATA)>
```

が与えられている場合、前記ステップ２－１で生成されるＡＳＮ．１抽象構文（型）は、

```
A ::= CHOICE {
   b   B,
   c   C }
B ::= NULL
C ::= NULL
```

となる。このようなＤＴＤに従う符号化対象のＸＭＬデータとして、

```
〈a〉
   〈b〉10〈/b〉
〈/a〉
```

を例示すれば、前記ステップ２－２で分離される要素（ＸＭＬデータの要素名と構造）は、

```
〈a〉
   〈b /〉
〈/a〉
```

となる。この要素から前記ステップ２－３で生成されるＡＳＮ．１抽象構文（値）は、

```
a A ::= b:NULL
```

となる。

【００５１】ＰＥＲでは、ｃｈｏｉｃｅ型の値は、まず選択された構成要素のインデクス（０ベース）が符号化され、次にその構成要素の値が符号化される。ここではｂが選択されているため、インデクスは０である。また、ｂの値はｎｕｌｌ　である。従って、ａの値は、

$0XXXXXXX_{(2)}$

のようになる。ここで、Ｘは、８ビットの倍数にするために付加されたパディング・ビットを表わす。このようにして、ＡＳＮ．１転送構文が生成される。なお、圧縮テキストの生成および圧縮テキストとＡＳＮ．１転送構文の併合は前記“４．「，」オペレータを含む場合”と同様である。

【００５２】６．「？」オペレータを含む場合

ここでは、「？」オペレータを含むＤＴＤと、それに従うＸＭＬデータを例にとる。以下のＤＴＤ、

```
<!ELEMENT a (b?,c)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (#PCDATA)>
```

が与えられている場合、前記ステップ２－１で生成されるＡＳＮ．１抽象構文（型）は、

```
A ::= SEQUENCE {
   id0   SEQUENCE {
      b   B OPTIONAL },
      c     C }
B ::= NULL
C ::= NULL
```

となる。このようなＤＴＤに従う符号化対象のＸＭＬデータとして、

```
〈a〉
   〈b〉10〈/b〉
   〈c〉20〈/c〉
〈/a〉
```

を例示すれば、前記ステップ２－２で分離される要素（ＸＭＬデータの要素名と構造）は、

```
〈a〉
   〈b /〉
   〈c /〉
〈/a〉
```

となる。この要素から前記ステップ２－３で生成されるＡＳＮ．１抽象構文（値）は、

```
a A ::= {
   id0   {
      b   NULL }
      c   NULL }
```

となる。

【００５３】ＰＥＲでは、ｓｅｑｕｅｎｃｅ型の構成要素が少なくとも１つ「ＯＰＴＩＯＮＡＬ」と指定されている場合に、各構成要素の値が符号化される前に、どの構成要素が存在しているかを表わすビット列が付加される。ビットは、構成要素が存在する場合に１、存在しない場合に０となる。ここではｂだけが「ＯＰＴＩＯＮＡＬ」と指定されており、それが存在しているため、まずビット列１　が付加される。次にｂとｃの値が符号化されるが、それらは共にｎｕｌｌであるため、ａの値は、

$1XXXXXXX_{(2)}$

のようになる。このようにして、ＡＳＮ．１転送構文が生成される。なお、圧縮テキストの生成および圧縮テキストとＡＳＮ．１転送構文の併合は前記“４．「，」オペレータを含む場”と同様である。

【００５４】７．「＊」オペレータを含む場合

ここでは、「＊」オペレータを含むＤＴＤと、それに従うＸＭＬデータを例にとる。以下のＤＴＤ、

```
<!ELEMENT a (b*)>
<!ELEMENT b (#PCDATA)>
```

が与えられている場合、前記ステップ２－１で生成されるＡＳＮ．１抽象構文（型）は、

```
A ::= SEQUENCE OF B
```



```
B ::= NULL
```

となる。このようなＤＴＤに従う符号化対象のＸＭＬデータとして、

```
〈a〉
   〈b〉10〈/b〉
   〈b〉20〈/b〉
〈/a〉
```

を例示すれば、前記ステップ２－２で分離される要素（ＸＭＬデータの要素名と構造）は、

```
〈a〉
   〈b /〉
   〈b /〉
〈/a〉
```

となる。この要素から前記ステップ２－３で生成されるＡＳＮ．１抽象構文（値）は、

```
a A ::= {
   NULL,
   NULL }
```

となる。

【００５５】ＰＥＲでは、ｓｅｑｕｅｎｃｅ－ｏｆ型の値は、まずその構成要素の数が符号化され、次に各構成要素の値がその順番で符号化される。ここでは構成要素の数は２であり、まずその値が符号化される。次に各構成要素の値が符号化されるが、それらはすべてｎｕｌｌであるため、ａの値は、$00000010_{(2)}$のようになる。このようにして、ＡＳＮ．１転送構文が生成される。なお、圧縮テキストの生成および圧縮テキストとＡＳＮ．１転送構文の併合は前記“４．「，」オペレータを含む場合”と同様である。

【００５６】８．混在内容を含む場合

ここでは、混在内容を含むＤＴＤと、それに従うＸＭＬデータを例にとる。以下のＤＴＤ、

```
<!ELEMENT a (#PCDATA|b)*>
<!ELEMENT b (#PCDATA)>
```

が与えられている場合、前記ステップ２－１で生成されるＡＳＮ．１抽象構文（型）は、

```
A ::= SEQUENCE OF CHOICE {
   txt   NULL,
   b     NULL }
```

となる。このようなＤＴＤに従う符号化対象のＸＭＬデータとして、

```
〈a〉
   xxx
   〈b〉10〈/b〉
   〈b〉20〈/b〉
〈/a〉
```

を例示すれば、前記ステップ２－２で分離される要素（ＸＭＬデータの要素名と構造）は、

```
〈a〉
  〈txt /〉
  〈b /〉
  〈b /〉
〈/a〉
```

となる。

【００５７】なお、上記において「ｘｘｘ」はテキストの内容を示し、「ｔｘｔ」は要素がテキストであることを示す。この要素から前記ステップ２－３で生成されるＡＳＮ．１抽象構文（値）は、

```
a A ::= {
    txt:NULL、
    b:NULL、
    b:NULL }
```

となる。

【００５８】ｓｅｑｕｅｎｃｅ－ｏｆ型の値とｃｈｏｉｃｅ型の値をＰＥＲで符号化する方法は、前述の通りである。ここでは、構成要素の数は３である。また、構成要素としてｔｘｔ、ｂ、ｂがその順番で選択されているため、インデクスは０、　１、１となる。それらの値はすべてｎｕｌｌであるため、ａの値は、

00000011(2)

011XXXXX(2)

のようになる。このようにして、ＡＳＮ．１転送構文が生成される。なお、圧縮テキストの生成および圧縮テキストとＡＳＮ．１転送構文の併合は前記“４．「，」オペレータを含む場合”と同様である。

【００５９】（実施の形態２）前記実施の形態１では、ＸＭＬデータは要素とテキストだけから構成され、その他のもの（例えば、属性や処理命令）は別の方法で管理されると仮定して説明を行った。本実施の形態では、属性や処理命令等が含まれる場合の処理の一例を説明する。すなわち、それら属性等を特別な要素で表現し、ＸＭＬデータの中に埋め込む方法を例示する。この場合、文法も変更する必要がある。なお、本実施の形態の例に関わらず、属性等をＸＭＬデータから分離して、ＸＰｏｉｎｔｅｒなどと共に別に格納する方法を採用することも可能である。このような属性等を特別な要素としてＸＭＬデータに埋め込む方策は、前記実施の形態１の処理の前処理および後処理として把握することが可能である。これにより広い範囲のＸＭＬデータを扱えるようになる。以下では、そのような前処理の例として、ＤＴＤで定義される項目をＸＭＬデータの中に埋め込む方法を説明する。

【００６０】９．１　前処理システムと前処理の概要

図３は、本実施の形態の前処理システムの一例をその機能について示したブロック図である。本実施の形態の前処理システムは、ＤＴＤ変換装置９－１、ＸＭＬデータ変換装置９－２、符号化装置９－３を有する。

【００６１】ＤＴＤ変換装置９－１は、一般のＤＴＤを

要素以外の項目についても考慮したＤＴＤ’に変換する。ＸＭＬデータ変換装置９－２は、一般のＤＴＤに従うＸＭＬデータをＤＴＤ’に従うＸＭＬデータ’に変換する。符号化装置９－３は、実施の形態１の符号化システムである。

【００６２】本実施の形態の前処理方法は、装置９－１によるＤＴＤの変換および装置９－２によるＸＭＬデータの変換の各ステップを有する。各々変換後、実施の形態１の符号化処理を行うことにより符号化ＸＭＬデータが生成される。

【００６３】９．２　ＤＴＤの変換処理

ＤＴＤには、要素型宣言以外に属性リスト宣言、エンティティ宣言、記法宣言の３つの宣言が含まれうる。

【００６４】エンティティ宣言で定義されるエンティティには、パース対象エンティティ、パース対象外エンティティ、パラメータ・エンティティがある。パース対象エンティティは、テキストや属性値など、あらゆる場所で参照されうる。このエンティティは、単純に展開する。また、パース対象外エンティティは、属性値でしか参照されえない。従って、属性を処理できれば十分である。一方、パラメータ・エンティティは、ＤＴＤの中でしか参照されえない。従って、パラメータ・エンティティは本実施の形態では考慮しない。

【００６５】記法宣言で定義される記法は、属性値でしか参照されえない。従って、属性を処理できれば十分である。

【００６６】以上の議論から、ＤＴＤに対する前処理は、要素型宣言と属性リスト宣言に対するそれに帰着される。

【００６７】属性リスト宣言では、ある要素に付与される属性と、その属性の取りうる値が定義される。また、必要であれば、デフォルト値も定義される。属性は、「ＲＥＱＵＩＲＥＤ」であるか、「ＩＭＰＬＩＥＤ」であるか、デフォルト値が定義されているか、それは「ＦＩＸＥＤ」であるかで、要素に対する付与のされ方が変わる。従って、属性は原則として要素で表現することにするが、その定義に従って表現を多少変えることが好ましい。

【００６８】属性リスト宣言に従って要素型宣言を変更する規則は、以下のとおりである。

（１）属性は、要素（以下、属性要素）で表現する。要素名は、属性名から一意に決定できるようにする。例えば、“親の要素名”＋“＿”＋“属性名”のようにする。ただし、要素名がすでに使用されていないことに注意する。以下では、この命名規則に従うことにする。

（２）属性値はすべて「ＣＤＡＴＡ」として扱い、属性要素に含める。

（３）属性要素は、親要素の子要素の先頭に挿入する。

（４）「ＲＥＱＵＩＲＥＤ」属性は、要素で表現する。

（５）「ＩＭＰＬＩＥＤ」属性は、「？」オペレータが

適用された要素で表現する。

（６）デフォルト値が定義されている属性は、「？」オペレータが適用された要素で表現する。

（７）「ＦＩＸＥＤ」デフォルト値が定義されている属性は、無視する。

【００６９】以下具体例をあげる。例えば、ＤＴＤとして、

```
〈!ELEMENT a (b,c)〉
〈!ATTLIST a ▽ CDATA #REQUIRED
             x ID #IMPLIED
             y (0|1|2) "0"
             z CDATA #FIXED "abc" 〉
```

が与えられたとき、ＤＴＤ’は、

```
<!ELEMENT a (a_w,a_x?,a_y?,b,c)>
<!ELEMENT a_w (#PCDATA)>
<!ELEMENT a_x (#PCDATA)>
<!ELEMENT a_y (#PCDATA)>
```

となる。

【００７０】９．３　ＸＭＬデータの変換処理

ここでは、上記のＤＴＤに従うＸＭＬデータを上記ＤＴＤ’に従うＸＭＬデータ’に変換する。基本的には、要素に付与されている属性を属性要素に変換し、その属性要素をその要素の子要素の先頭に挿入すればよい。ただし、属性にデフォルト値が定義されており、属性値がそれと一致するのであれば、属性要素には変換しない。

【００７１】例えば、９．２におけるＤＴＤに従うＸＭＬデータとして、

```
〈a ▽= "xyz"  y= "0"  z= "abc" 〉
   〈b〉...〈/b〉
   〈c〉...〈/c〉
〈/a〉
```

を例示すれば、ＸＭＬデータ’は、

```
〈a〉
   〈a_▽〉xyz〈/a_▽〉
   〈b〉...〈/b〉
   〈c〉...〈/c〉
〈/a〉
```

に変換される。ここで、ａ要素にはｘ属性が付与されていないため、ａ＿ｘ要素は現れていない。また、ｙ属性の値はデフォルト値と一致するため、ａ＿ｙ要素も現れていない。さらに、ｚ属性は「ＦＩＸＥＤ」デフォルト値が定義されているため、削除されている。

【００７２】９．４　後処理システムと後処理の概要

図４は、本実施の形態の後処理システムの一例をその機能について示したブロック図である。本実施の形態の後処理システムは、ＤＴＤ変換装置９－４、復号化装置９－５、ＸＭＬデータ変換装置９－６を有する。

【００７３】ＤＴＤ変換装置９－４はＤＴＤ変換装置９－１と同様である。復号化装置９－５は、実施の形態１

の復号化システムである。ＸＭＬデータ変換装置９－６は、ＤＴＤ’に従うＸＭＬデータ’からＤＴＤに従うＸＭＬデータに変換する。

【００７４】本実施の形態の後処理方法は、装置９－４によるＤＴＤの変換および装置９－６によるＸＭＬデータの変換の各ステップを有する。実施の形態１の復号化処理の後、これらステップを実行する。装置９－４によるＤＴＤの変換は９．２と同様であり、装置９－６によるＸＭＬデータの変換は９．３の変換を逆に行うことにより実行できる。よって、詳細な説明は省略する。なお、図中破線矢印は、必要があればＤＴＤを参照することを意味する。属性要素を識別でき、かつその要素名から属性名を一意に決定できるのであれば、ＤＴＤを参照する必要はない。また、後処理は前処理がいかに行われたかに依存するので、前記前処理が異なる場合に本実施の形態の後処理もそれに併せて変更されることは勿論である。

【００７５】１０．実施の形態の効果

上記した実施の形態１、２の符号化方法を用いれば、ＸＭＬデータを効率良く圧縮することが可能である。以下、ＸＣｏｍｐとの比較において、本実施の形態の効果を説明する。

【００７６】ＸＭＬデータを符号化するときに、ＤＴＤから一意に決定される情報は符号化しないという点で、本手法とＸＣｏｍｐは同じである。しかし、ＸＣｏｍｐは、ＸＭＬデータがある特定の構造をしている場合に、圧縮効率が悪くなる。具体的には、要素に「？」オペレータや「＊」オペレータ（「＋」オペレータを含む）が適用された場合である。

【００７７】１０．１　「？」オペレータ

本実施の形態の手法は要素が存在するかどうかをビット列で表現するのに対して、ＸＣｏｍｐは存在する要素（選択肢）に付与されたインデクスを羅列する。この差は、「？」オペレータが適用された要素がいくつか連続し、それらがすべて存在する場合に顕著に現れる。

【００７８】例えば、ＤＴＤ：

```
<!ELEMENT a (b?,c?)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (#PCDATA)>
```

に従うＸＭＬデータ：

```
〈a〉
   〈b〉xxx〈/b〉
   〈c〉yyy〈/c〉
〈/a〉
```

を考える。このＸＭＬデータには、要素ｂと要素ｃが共に存在している。このＸＭＬデータは、本手法では１１（２）に符号化される。ただし、簡単のため、パディング・ビットは付加していない。

【００７９】一方、ＸＣｏｍｐでは、オートマトン：

25

	S1	S2	S3	S4
S1	-	b/1	c/2	ε/3
S2	-	-	c/1	ε/2
S3	-	-	-	ε/ε
S4	-	-	-	-

が作成されたとすると、インデクス列１１が得られる。このインデクス列を０ベースに変換し、必要最小限のビット数で符号化すると、符号０００（２）が得られる。本手法による符号とＸＣｏｍｐによるそれを比較すると、本手法のほうが１ビットだけ短く符号化できることが分かる。

【００８０】表１に、「？」オペレータが適用された要素がいくつか連続し、それらがすべて存在する場合に、それらを符号化するのに必要なビット数を、本手法とＸＣｏｍｐで比較したものを示す。表１から、本手法はＸＣｏｍｐと同じか、それよりも効率よく符号化できることが分かる。一般には、「？」オペレータが適用された要素がｎ個連続する場合に、ＸＣｏｍｐはＯ（ｎｌｏｇｎ）ビットを必要とするのに対して、本手法はＯ（ｎ）ビットで十分である。

【００８１】

【表１】

	XComp(従来技術)			本発明
"？"の数	選択肢数	ビット数	トータルビット数	ビット数
1	2	1	1	1
2	3、2	2、1	3	2
3	4、3、2	2、2、1	5	3
4	5、4、3、2	3、2、2、1	8	4
5	6、5、4、3、2	3、3、2、2、1	11	5

【００８２】１０．２　「＊」オペレータ

本実施の形態の手法は存在する要素の数を符号化するのに対して、ＸＣｏｍｐは存在する要素（選択肢）に付与されたインデクスを羅列する。この差は、「＊」オペレータが適用された要素がいくつも存在する場合に顕著に現れる。

【００８３】例えば、ＤＴＤ：

<!ELEMENT a (b*)>

<!ELEMENT b (#PCDATA)>

に従うＸＭＬデータ：

```
<a>
  <b>xxx</b>
  <b>yyy</b>
  ...
  <b>zzz</b>
</a>
```

を考える。ここで、要素ｂは８回出現しているとする。このＸＭＬデータは、本手法では００００１０００（２）に符号化される。

【００８４】一方、ＸＣｏｍｐでは、オートマトン：

	S0	S1
S0	b/1	ε/2
S1	-	-

が作成されたとすると、インデクス列１１１１１１１１２が得られる。このインデクス列を０ベースに変換し、必要最小限のビット数で符号化すると、符号０００００００００１（２）が得られる。本手法による符号とＸＣｏｍｐによるそれとを比較すると、本手法のほうが１ビットだけ短く符号化できることが分かる。

【００８５】図５に、「＊」オペレータが適用された要素がいくつも存在する場合に、それらを符号化するのに必要なビット数を、本手法とＸＣｏｍｐで比較したものを示す。同図から、要素が８つ以上存在する場合に、本手法はＸＣｏｍｐよりも効率よく符号化できることが分かる。本手法は、一般にはＸＣｏｍｐと同じオーダのビット数を必要とするが、実質的にはそれよりも少ないビット数で十分である。

【００８６】以上、本発明者によってなされた発明を発明の実施の形態に基づき具体的に説明したが、本発明は前記実施の形態に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能である。

【００８７】たとえば、本発明の手法は、複数のオペレータを組み合わせて適用しても良い。内容モデルで使用されうるオペレータには、「，」、「｜」、「？」、「＊」（「＋」　を含む）の４つがある。従って、適用順序を考慮に入れると、それらの組み合わせは１６通りある。各組み合わせに対して、前記実施の形態と同様にＸＭＬデータをＡＳＮ．１抽象構文に変換することができる。よって、本発明を利用してあらゆるオペレータの組合せを含むＸＭＬデータを符号化できる。

【００８８】また、前記実施の形態で説明した手法のポイントの１つは、文法やそれに従うＸＭＬデータのＡＳＮ．１抽象構文における表現を定義したことである。しかし、それとは別の表現も考えられる。

【００８９】たとえば、「？」オペレータは、前記実施の形態ではｓｅｑｕｅｎｃｅ型とキーワード「ＯＰＴＩＯＮＡＬ」の組み合わせで表現した。具体的には、ＤＴＤ：

<!ELEMENT a (b?,c)>

は、ＡＳＮ．１抽象構文：

27

```
A ::= SEQUENCE {
    id0   SEQUENCE {
    b   B OPTIONAL },
    c   C }
```

に変換される。

【００９０】しかし、先のＤＴＤはＡＳＮ．１抽象構文：

```
A ::= SEQUENCE {
    b   B OPTIONAL,
    c   C }
```

に変換されてもよい。このように変換されるほうが、ＢＥＲやＤＥＲで符号化する場合に符号が短くなるため都合がよい。

【００９１】ただし、この方法では、以下のＤＴＤ：

<!ELEMENT a (b?)>

<!ELEMENT a (b?|c)>

のような場合にうまく機能しないので、実施の形態のように表現するほうが妥当である。

【００９２】また、前記実施の形態では、オペレータは何からの型で表現される。例えば、「，」オペレータは

```
A ::= SEQUENCE {
    b   B }
```

と表現できる。

【００９４】この表現法では、ｓｅｑｕｅｎｃｅ型で統一的に表現することで、見た目には分かりやすくなる。また、オペレータをｓｅｑｕｅｎｃｅ型でラップすることに相当するので、複数のオペレータを組み合わせてもうまく機能することが分かる。ただし、ＢＥＲやＤＥＲでは常に型が符号化されるため、型を挿入すればするほど符号は長くなる。従って、必要以上の型を挿入するべきではなく、その理由から前記実施の形態の手法で表現するほうが妥当である。

【００９５】

【発明の効果】本願で開示される発明のうち、代表的なものによって得られる効果は、以下の通りである。すなわち、ＸＭＬデータの符号化（圧縮）効率を高めることが可能になり、また、属性等要素以外の記述を含めたＸＭＬデータの符号化（圧縮）が可能になる。これにより、ＸＭＬデータの転送による通信負荷を軽減し、また、ＸＭＬデータのストレージ容量を少なくできる。

【図面の簡単な説明】

【図１】本発明の一実施の形態である符号化システムの一例をその機能について示したブロック図である。

【図２】本発明の一実施の形態である復号化システムの一例をその機能について示したブロック図である。

【図３】本発明の一実施の形態である前処理システムの一例をその機能について示したブロック図である。

【図４】本発明の一実施の形態である後処理システムの一例をその機能について示したブロック図である。

【図５】本実施の形態の符号化手法の符号化率をＸＣｏｍｐとの比較において示した図である。

【符号の説明】

２－１…構文（型）生成装置、２－２…分離装置、２－３…構文（値）生成装置、２－４…転送構文生成装置、２－５…圧縮装置、２－６…併合装置、３－１…構文（型）生成装置、３－２…転送構文復号装置、３－３…抽象構文復号装置、３－４…併合装置、３－５…分離装置、３－６…解凍装置、９－１…ＤＴＤ変換装置、９－２…ＸＭＬデータ変換装置、９－３…符号化装置、９－４…ＤＴＤ変換装置、９－５…復号化装置、９－６…ＸＭＬデータ変換装置。

28

ｓｅｑｕｅｎｃｅ型で表現され、「｜」オペレータはｃｈｏｉｃｅ型で表現される。一方、オペレータの適用をプロダクションと見なし、すべてｓｅｑｕｅｎｃｅ型で表現するという方法も採れる。その場合に、ｓｅｑｕｅｎｃｅ　型で表現されていない「｜」オペレータ、「＊」オペレータ（「＋」オペレータを含む）およびオペレータなしは、次のように表現できる。

【００９３】すなわち、<!ELEMENT a(b|c)>は、

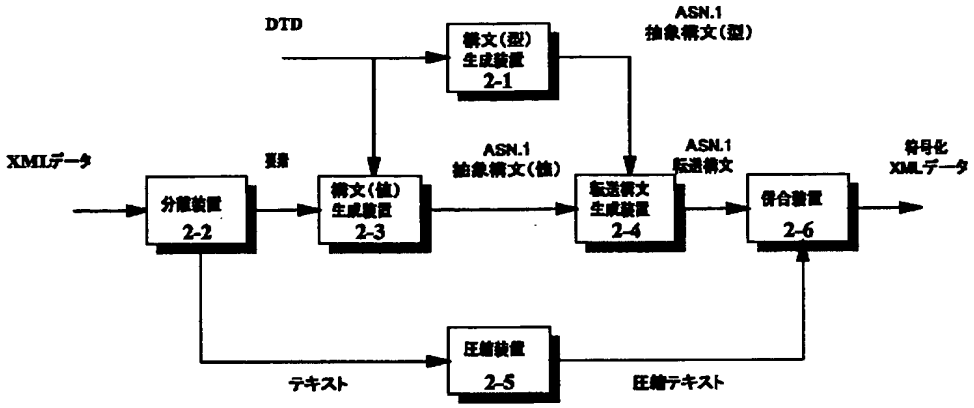
```
A ::= SEQUENCE {
    id0   CHOICE {
    b   B,
    c   C } }
```

と表現できる。また、<!ELEMENT a (b*)>は、

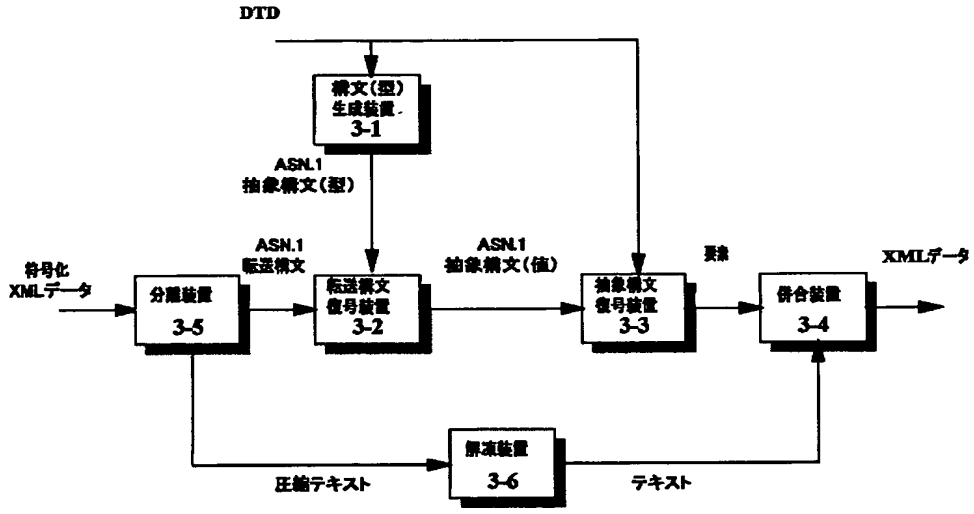
```
A ::= SEQUENCE {
    b   SEQUENCE OF B }
```

と表現できる。また、<!ELEMENT a (b)>は、

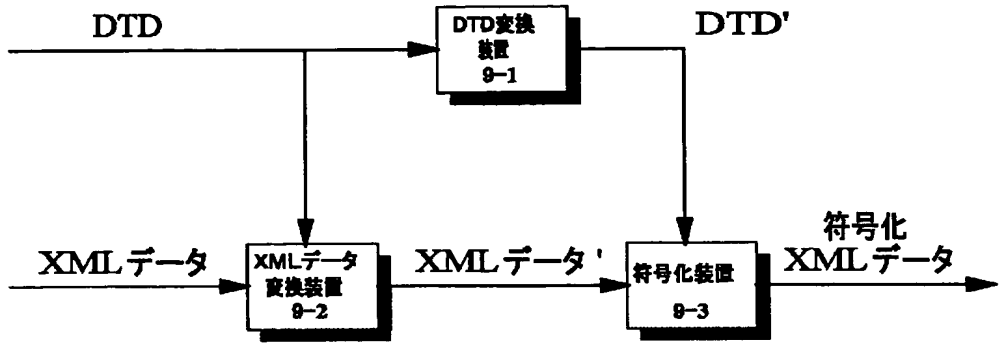
【図１】


DTD
構文(型)
生成装置
2-1
ASN.1
抽象構文(型)
XMLデータ
要素
分離装置
2-2
構文(値)
生成装置
2-3
ASN.1
抽象構文(値)
転送構文
生成装置
2-4
ASN.1
転送構文
併合装置
2-6
符号化
XMLデータ
圧縮装置
2-5
テキスト
圧縮テキスト

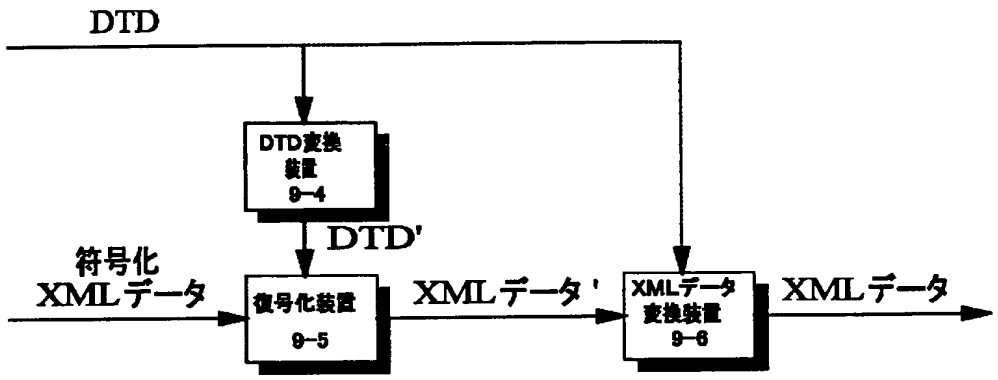
【図２】


DTD
構文(型)
生成装置
3-1
ASN.1
抽象構文(型)
符号化
XMLデータ
ASN.1
転送構文
分離装置
3-5
転送構文
復号装置
3-2
ASN.1
抽象構文(値)
抽象構文
復号装置
3-3
要素
併合装置
3-4
XMLデータ
圧縮テキスト
解凍装置
3-6
テキスト

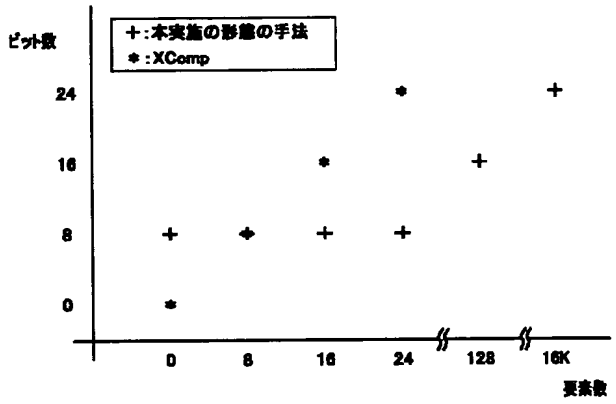
【図３】


DTD
DTD変換
装置
9-1
DTD'
XMLデータ
XMLデータ
変換装置
9-2
XMLデータ'
符号化装置
9-3
符号化
XMLデータ

【図４】



【図５】



フロントページの続き

(51)Int.Cl.[7]	識別記号	F I	ﾃｰﾏｺｰﾄﾞ(参考)
// Ｇ０６Ｆ　17/21	５７０	Ｇ０６Ｆ　17/21	５７０Ｇ

(72)発明者　今村　剛
神奈川県大和市下鶴間1623番地14　日本アイ・ビー・エム株式会社　東京基礎研究所内

Ｆターム(参考)　5B009 SA07
5B082 GA01
5B089 HB10 KA08 KH04 KH28
5J064 AA02 BA11 BC02 BD03